AMENDMENTS TO THE SPECIFICATION

Please replace the paragraph beginning on page 10, line 4 with the following paragraph rewritten in amendment format:

If, as illustrated at step **62,** the identified run length (j) is vertically adjacent to a run length (k) from a previous row, then the current run length's parent data structure (j's parent data structure) is populated with a reference to the vertically adjacent run length (k), as shown at 64. The algorithm then tests k's child data structure at step 66 to determine if it is currently empty (containing a null value). If a null value is found, the algorithm populates k's child data structure with a reference to scan length j, as illustrated at 68. If k's child data structure is not currently empty (in other words if it contains a reference to another child (s), and if the other child ~~(s) does not already have a sibling~~, the algorithm detects this at step 70 and then populates [[s's]] j's sibling data structure with [[j]] the value stored as k's child (s), as shown at **72.** Thus steps 66-72 essentially test whether the child of k is null. If so, it sets j to the child of k. Otherwise it sets j as the sibling of the child of k ~~(s) in the case where s has no other siblings~~.

Please replace the paragraph beginning on page 10, line 16 with the following paragraph rewritten in amendment format:

In some instances the algorithm will detect that the child ~~(s)~~ already has a sibling. In this case the algorithm ~~checks whether s's sibling has a sibling, and so on.~~ sets the child as the sibling's sibling, and so on. This is shown in steps 74 and 76. ~~When a null reference is finally reached, the null reference is replaced with a reference to i as~~

~~shown at step 78~~. When siblings are identified the algorithm sets their respective sibling pointers to each other as shown at step 72 and 78.